Remarks

Entry of the amendments, reconsideration of the application, as amended, and allowance of all pending claims are respectfully requested. Claims 1-69 remain pending.

In the Office Action dated July 29, 2003, claims 1-3, 9, 12, 15-16, 22-24, 30, 33, 36-37, 43-45, 49-50, 57, 60 & 63-64 are rejected under 35 U.S.C. §102(b) as being anticipated by Lynch (U.S. Patent No. 5,829,031); claims 4-5, 25-26 & 52-53 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Kahle et al. (U.S. Patent No. 6,574,712; hereinafter, "Kahle"); claims 6-7, 27-28 & 54-55 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Kahle, and further in view of Ryan (U.S. Patent No. 5,367,656); claims 8, 14, 29, 35, 56 & 62 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Lopez-Aguado et al. (U.S. Patent No. 6,317,810; hereinafter, "Lopez-Aguado"); claims 10, 13, 31, 34, 59 & 61 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Ryan; claims 11, 32 & 59 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Ryan, and further in view of Lopez-Aguado; claims 17, 20, 38, 41, 47-48, 65 & 68 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Mason, Jr. (U.S. Patent No. 5,884,098); claims 18, 39 & 66 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Mason Jr., and further in view of Ryan; claims 19, 40 & 67 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Mason, Jr., and further in view of Lopez-Aguado; and claims 21, 42 & 69 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lynch in view of Mason, Jr., and further in view of Ryan. Applicants respectfully, but most strenuously, traverse these rejections to any extent deemed applicable to the claims presented herewith.

Independent claims 1, 12, 17, 22, 33, 38, 43, 45, 47, 49, 60 & 65 are amended to more clearly point out and distinctly claim certain aspects of applicants' invention. In particular, claims 1, 22, 43 & 49 are amended to characterize the pattern of requests detected as being based on one or more user-defined attributes of the multiple files. For support, see, e.g., p. 8, lines 6-10 and p. 11, lines 4-11 of applicants' specification. Further, claims 12, 33, 45 & 60 are amended to include controlling a rate of prefetching that occurs subsequent to determining that prefetching is to occur (see, e.g., FIG. 6 and p. 13, lines 24-27 of applicants' specification for support). Still further, claims 17, 38, 47 & 65 are amended to characterize the detected pattern of requests as being based on directory entries of the multiple files being within the directory block (see, e.g., FIGs. 3 & 4; p. 7, line 26 – p. 8, line 5; and p. 11, lines 4-17 of applicants' specification). Dependent claims 3, 24, 45 & 51 are also amended herein. Support for these amendments can be found throughout applicants' specification (see, e.g., p. 10, line 12 – p. 11, line 17). No new matter is believed added to the application by the amendments presented herewith.

Applicants recite a management technique for prefetching of data of files (e.g., claim 1, as amended herein). The technique includes, for instance, (1) detecting a pattern of requests for data of multiple files, wherein the pattern is based on one or more user-defined attributes of the multiple files; and (2) prefetching data of a plurality of files, in response to the detecting indicating the pattern.

In another embodiment (e.g., claim 12, as amended herein), applicants recite a technique of managing the prefetching of data that includes, for example, (1) controlling, subsequent to determining that prefetching of data is to occur, a rate at which data of a plurality of files is prefetched by pacing the prefetching based upon requests for data; and (2) prefetching the data of the plurality of files in response to the controlling.

Advantageously, applicants' recited prefetch management technique provides for efficient access of inodes (i.e., data structures that include meta data) by, for instance, prefetching inodes at substantially the same rate at which an application requests the inodes.

Thus, in applicants' invention, the detected pattern of data requests is based on user-defined attributes of multiple files. Further, the rate of data prefetching is controlled by pacing the prefetching based upon data requests. Applicants respectfully submit that at least these aspects are not taught or suggested by Lynch.

Lynch describes a microprocessor with a heuristic processing unit that detects a pattern of data accesses by a group of instructions, wherein the pattern is representative of memory addresses generated by instructions and constant values stored within addressing registers (i.e., values used to form addresses of memory operations) (see Abstract; col. 8, lines 34-35; and col. 8, line 66 – col. 9, line 9). Upon detection of this memory address-based pattern, data is prefetched into a data cache residing on the microprocessor, according to the detected pattern (see FIG. 1; col. 8, lines 35-38; and col. 9, lines 13-15). This pattern detection and prefetching based on memory addresses is quite different from applicants' claimed invention as presented herewith.

For example, applicants recite detecting a pattern of requests for data of multiple files, wherein the pattern is based on one or more user-defined attributes of the multiple files (e.g., claim 1). These user-defined attributes of the multiple files can include, for instance, a user-defined location of the files (e.g., a single user-defined directory of a file system). In one example of the present invention, a counter associated with file accesses from a single directory is compared to a cache miss threshold. When the counter exceeds the cache miss threshold value, it indicates that upcoming requests are likely to request meta data associated with all (or a large subset) of the files of the directory, and that the meta data to be requested is not already cached. In this case, the application would benefit from a prefetch of meta data from that same directory (see specification, page 11, lines 4-11). Thus, in this example, the pattern of requests for meta data of multiple files is based on a user-defined attribute (e.g., the files being stored within a single user-defined directory).

In contrast, the microprocessor detecting data access patterns in Lynch does not use and has no knowledge of user-defined attributes of files. Instead of using knowledge that data accesses are associated with files located within, for example, a single directory of a file system, Lynch is limited to detecting a pattern of data accesses based on memory addresses, which are not user-defined. At col. 8, line 66 – col. 9, line 9, Lynch describes, with reference to FIG. 3 thereof, the group of instructions that indicate the pattern of data accesses:

> Heuristic processing unit 36 begins detection of the group of instructions at step 50 by capturing a register used as an addressing register (i.e., a register storing a value used to form the address of a memory operation). Subsequent instructions are analyzed by heuristic processing unit 36 according to steps 52, 54, and 56. If the register is found to be the target of another instruction (step 52), and the instruction modifies the register by a constant value (step 54), and the register is used in a subsequent instruction to form an address (step 56), then heuristic processing unit 36 notifies data cache 24 to begin prefetching (step 58).

Thus, the data access pattern detected in Lynch is based on the formation of a memory address. This memory address and its formation are not user-defined, which differs from a pattern of data requests based on one or more user-defined attributes of multiple files, as recited by the claims presented herewith.

In the Office Action, col. 8, lines 32-45 of Lynch is cited as disclosing the detecting of a pattern of requests for data of multiple files and prefetching data of a plurality of files, in response to the detecting indicating the pattern. This section of Lynch generally describes detecting a data access pattern and prefetching additional data according to the detected pattern, which was described in more detail above. As noted, Lynch discloses detecting a memory address-based pattern, which is not based on one or more user-defined attributes of multiple files, as recited by the present invention.

Based on the foregoing, applicants respectfully submit that Lynch does not teach, or even suggest, at least applicants' recited feature of detecting a pattern of requests where the pattern is based on one or more user-defined attributes of multiple files. Further, applicants respectfully submit that the other applied art does not teach or suggest the above-described recited feature of the present invention. Thus, applicants respectfully request reconsideration and withdrawal of the anticipation rejection of independent claims 1, 22, 43 & 49.

Relative to the anticipation rejection of independent claims 12, 33, 45 & 60, applicants respectfully submit that the timing of prefetching in Lynch is very different from the prefetching management technique of the present invention.

For example, applicants recite controlling a rate at which data of a plurality of files is prefetched by pacing the prefetching based upon requests (e.g., claim 12). This control of the rate is done subsequent to determining that prefetching of data is to occur (e.g., claim 12). That is, the prefetch management technique of the present invention first determines that prefetching is to occur (e.g., by detecting a pattern of requests for data of multiple files) and then decides how quickly to prefetch (i.e., by controlling the rate at which data is prefetched). This rate-related decision is significant because simply prefetching as quickly as possible can negate certain benefits of prefetching. For instance, prefetching inodes too fast can fill up a cache prematurely and cause some previously prefetched inodes to be thrown out of the cache before the application has a chance to access them (see, e.g., specification, page 14, lines 14-23).

Thus, to retain certain benefits of prefetching data, the present invention controls the rate at which data is prefetched, subsequent to determining that prefetching is to occur, by pacing the prefetching based upon requests for data. For example, after the present invention decides to prefetch inodes for the next 100 files, it may be advantageous to control the rate of the prefetching by, for instance, prefetching the first ten inodes, waiting for an amount of time, prefetching the next ten, waiting again, etc. This control of the prefetch rate can, for example, allow the rate of prefetching inodes to substantially match the speed at which the application is requesting access to inodes.

In contrast, Lynch does not describe controlling a rate at which data is prefetched at all. Instead, Lynch simply describes the initiation of prefetching as occurring when the above-described data access pattern is detected (col. 8, lines 32-45). Applicants respectfully submit that determining when prefetching is initiated is clearly different from controlling a rate at which data is prefetched, as recited by the claims presented herewith. Not only does Lynch fail to describe or suggest such control of a prefetch rate, it also does not teach or suggest that such a rate is controlled subsequent to determining that prefetching is to occur, or that the rate is controlled by pacing the prefetching based upon requests for data, as claimed by the present invention.

The Office Action cites col. 8, lines 32-45 of Lynch as teaching control of the prefetching of data by pacing at least the initiating of the prefetching based upon requests for data, and further states that the pacing is based upon the detection of a pattern of requests. As noted, this section describes Lynch's determination of when prefetching is started, but does not disclose or suggest controlling the rate of prefetching.

For the above reasons, it is respectfully submitted that Lynch does not teach, or even suggest, applicants' recited feature of controlling a rate at which data of a plurality of files is prefetched. Moreover, applicants respectfully submit that this feature is not described, taught or suggested by the other applied art. Therefore, applicants respectfully request reconsideration and withdrawal of the 35 U.S.C. §102(b) rejection of independent claims 12, 33, 45 & 60.

In yet another embodiment (e.g., claim 17, as amended herein), applicants recite a technique for managing the prefetching of inodes associated with files of a directory, where the directory comprises one or more directory blocks and each directory block has associated with it zero or more files. This technique includes, for instance, (1) detecting a pattern of requests for multiple inodes associated with multiple files of a directory block of the one or more directory blocks, wherein the pattern is based on directory entries of the multiple files being within the directory block; and (2) prefetching a plurality of inodes associated with the directory block, in response to detecting the pattern.

Thus, the present invention includes, for example, a pattern of requests for inodes associated with files of a directory block, wherein the pattern is based on directory entries of the files being within the directory block. These directory entries include, for instance, file names and pointers to the disk locations of file data and meta data (see 304, FIG. 3). Detecting this pattern based on directory entries being within the directory block is one example of detecting a pattern based on the multiple files being within the same directory.

Relative to the 35 U.S.C. §103(a) rejection of independent claims 17, 38, 47 & 65, applicants respectfully submit that the above-described claimed element that includes the pattern based on directory entries of the multiple files being within the directory block (e.g., claim 17) is very different from the teachings of Lynch and Mason, either alone or in combination.

For example, as noted above, Lynch discloses a microprocessor configured to detect a pattern of data accesses based on memory addresses. This memory address-based scheme of Lynch does not need any knowledge of what, if any, directory block includes directory entries of files associated with the data being accessed. The memory address patterns of Lynch are formed by incrementing or decrementing addresses with constant values (see, e.g., col. 9, lines 14-23), and are not related at all to a meta data request pattern based on files being within a single directory, let alone one based on directory entries of multiple files being within a directory block, as recited by the present invention.

The Office Action cites col. 8, lines 32-45 as disclosing detecting a pattern of requests for data of multiple files of a directory block. As described above, this section of Lynch describes a memory address-based pattern of data access that is not related to whether or not the data is associated with files, a directory, or a directory block. Even if Lynch suggests a directory or directory block associated with the data being accessed, it is clear that the pattern of data access is based on memory addresses (e.g., an address being repeatedly incremented by a constant value to obtain other addresses; see col. 9, lines 14-23), which differs from a pattern based on directory entries of the multiple files being within a directory block, as recited by the claims presented herewith.

Further, at page 3, lines 9-11 of the Office Action, it is stated that multiple files being associated with a single directory is disclosed in Lynch's system that inherently includes a page table/translation table for the main memory depicted in FIG. 5 thereof. Applicants respectfully traverse the equivalency alleged between applicants' recited directory and a page table/translation table. Applicants' directory reflects a structure of data in files that is visible to the user and defined according to the intent of the user (e.g., a user places data files related to one project in a single directory to reflect the project-based relationship among the files). In contrast, a page table or translation table is a structure internal to the system of Lynch that allows, for instance, the tracking of the location of data in memory at a given time. These internal table structures are not visible to the user, nor are they user-defined. Thus, the page table/translation table of Lynch is not descriptive or suggestive of applicants' directory, let alone of the recited directory comprising one or more directory blocks, wherein a pattern of requests for inodes associated with multiple files of a directory block of the one or more directory blocks is detected, and wherein the pattern is based on directory entries of the multiple files being within the directory block.

Mason fails to overcome the above-described deficiency of Lynch as applied to the present invention. Mason describes a disk drive array controller that allows prefetching of disk drive meta data used exclusively by the disk array controller and other disk subsystem components for the control and maintenance of the disk array system. The meta data in Mason includes parity information (e.g., parity blocks) (col. 1, lines 41-47). The prefetching of meta data in Mason is quite different from the inode prefetch management technique of applicants' claimed invention.

For example, applicants recite detecting a pattern of requests for multiple inodes associated with multiple files of a directory block, wherein the pattern is based on directory entries of the multiple files being within the directory block. In contrast, Mason's prefetch of meta data is based on the likelihood of one command following another, without regard to whether the meta data is associated with multiple files of a directory block. For example, since a write command (which involves parity blocks) often follows a read of a block, Mason prefetches a parity block into a cache to save time and enhance performance of input/output (I/O) processing (see col. 7, lines 54-63). This parity block reflects internal structure of the storage system in Mason, and is not user-defined or visible to the user. Thus, this parity block prefetching in Mason is based on internal system structure, rather than on files being within the same user-defined directory, let alone on directory entries of those files being within a directory block, as recited by the claims presented herewith.

The Office Action cites col. 4, lines 30-41; col. 7, lines 53-63; and col. 9, lines 32-39 of Mason as teaching prefetching of meta data. These sections of Mason describe a prefetch technique that includes prefetching meta data (e.g., parity information/blocks) into a back end cache coordinated with a front end cache, in parallel with other operations, and in conjunction with a write command because it is likely to follow a read of a block. Although prefetching of meta data is described in these sections of Mason, they do not teach or suggest prefetching in response to detecting the pattern described above. Again, it is the detection of the pattern based on the above-noted directory entries that is not taught or suggested by Mason.

Since both Lynch and Mason fail to teach or suggest applicants' claimed element of detecting a pattern of requests for multiple inodes associated with multiple files of a directory block, wherein the pattern is based on directory entries of the multiple files being within the directory block, applicants submit that the combination also fails to teach or suggest this claimed element. Further, it is respectfully submitted that the other applied art does not describe or suggest at least this recited feature of the present invention. Therefore, applicants respectfully request an indication of allowability for independent claims 17, 38, 47 & 65, and all claims that depend therefrom.

The dependent claims are patentable for the same reasons as the independent claims from which they directly or ultimately depend, as well as for their own additional features. For example, in dependent claim 3, applicants explicitly claim that the multiple files and the plurality of files described above relative to claim 1 are within a single directory. That is, the multiple files whose data is requested in the detected pattern described above, and the plurality of files whose data is prefetched in response to detecting the pattern are located within the same directory.

In contrast, none of the applied art teaches or suggests that multiple files whose data is requested in a pattern and the plurality of files whose data is prefetched in response to detecting the pattern are located within a single directory. For example, as noted above, Lynch prefetches data according to the pattern established by detecting a group of instructions that access a next memory address determined by, for instance, adding a constant value to a current memory address. There is no teaching or suggestion in Lynch that the group of instructions request access to data in files of a directory at all, or that the prefetched data is associated with files of a directory, let alone that such files, if they exist, are located in a single directory, as recited by the present invention.

In the Office Action, it is stated that the multiple files and the plurality of files being associated with a single directory is disclosed in Lynch's system that inherently includes a page table/translation table for the main memory depicted in FIG. 5 thereof. As noted above, applicants traverse the alleged equivalency between applicants' recited directory and a page table/translation table. Again, applicants' recited directory reflects a structure of data in files that is user-defined and visible to the user. In contrast, a page table or translation table is a structure internal to Lynch's system that allows, for instance, the tracking of the location of data in memory at a given time. These internal table structures are not visible to the user, nor are they user-defined. Moreover, applicants submit that the other applied art does not teach or suggest the multiple files and the plurality of files being within a single directory. Thus, applicants respectfully submit that claim 3 and other similar dependent claims are patentable over Lynch, alone or in combination with the other applied patents.

For all of the above reasons, applicants respectfully request an indication of allowability for all pending claims.

Should the Examiner wish to discuss this case with applicants' attorney, please contact applicants' attorney at the below listed number.

Respectfully submitted,

Blanche E. Schiller

Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

Dated: December 01, 2003.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579